HEVC 의 메모리 요구량 감소를 위한 시간적 움직임 벡터 절삭

임성창, 김휘용, 김종호, 이진호, 전동산, 최진수
한국전자통신연구원 방통융합미디어연구부
sclim@etri.re.kr

Temporal Motion Vector Clipping for Reducing Memory Requirement of HEVC

Sung-Chang Lim, Hui Yong Kim, Jongho Kim, Jinho Lee, Dong-San Jun, and Jin Soo Choi
Broadcasting and Telecommunications Convergence Media Research Department, ETRI

요 약

본 논문에서는 시간적 움직임 벡터 성분 값을 절삭하여 시간적 움직임 벡터를 저장하는데 필요한 메모리 요구량을 감소하는 방법에 대해서 제안한다. HEVC 에서 시간적 움직임 벡터는 움직임 병합 모드와 향상된 움직임 벡터 예측 방법에서 부호화 효율 향상을 위해서 사용되고 있다. 하지만 부호화해야 할 원본 영상의 공간적 해상도 및 비트 심도가 증가함에 따라 메모리에 저장할 시간적 움직임 벡터의 데이터뿐만 아니라 메모리 접근 대역폭도 크게 증가한다. 따라서 본 논문에서는 시간적 움직임 벡터 성분의 비트 심도를 조절한 후 메모리에 저장하는 방법에 대해서 제안한다. 제안하는 방법을 HM 2.0 에 구현하고 HEVC 표준화에서 사용되는 임의접근 및 저지연 실험 조건에서 실험했을 때, 평균 0.2%의 비트율 손실을 보이지만, 필요한 메모리 요구량을 약 반 정도 줄일 수 있다.

1. 서론

최근 HD 급 이상의 고해상도 영상 컨텐츠에 대한 수요가 다양한 영상 응용 분야에서 급속히 증가하고 있지만, 이러한 영상 컨텐츠를 표준이 완료된 지 오래된 H.264/AVC (Advanced Video Coding) 표준 [1]을 비롯한 기존 영상 부호화 표준을 사용해서는 현재 네트워크 대역폭 (Bandwidth)이나 저장매체의 크기 제한으로 인해 수요자가 원하는 수준의 영상 서비스를 제공하는데 한계가 있다. 이러한 기술적인 문제를 해결하고자 영상 부호화의 양대 표준화 단체인 MPEG (Moving Picture Experts Group)과 VCEG (Video Coding Experts Group)에서는 2010 년 1 월 일본 교토 회의에서 차세대 영상 부호화 표준 개발을 위해 JCT-VC (Joint Collaborative Team on Video Coding)라는 공동 협력팀을 구성하기로 합의했으며, H.264/AVC 표준의 2 배 가량의 부호화 효율 향상을 목표로 한 HEVC (High Efficiency Video Coding) 표준의 CfP (Call for Proposal)를 공표했다 [2]. 이후 제 5 차 JCT-VC 회의까지 진행되었으며, 다양한 크기를 가질 수 있는 부호화/복호화의 기본 단위, 35 가지 화면내 예측 방법, 향상된 움직임 벡터 예측 (Advanced Motion Vector Prediction) 방법, 움직임 병합 (Motion Merge) 방법, 적응적 루프 필터링 방법 등의 기술들이 HEVC 표준의 기술로 채택된 상태이다 [3].

H.264/AVC 에서는 시간적 다이렉트 모드 (Temporal Direct Mode)와 공간적 다이렉트 모드 (Spatial Direct Mode)에서 움직임 벡터 유도와 부호화/복호화 대상 영역이 정적인 영역인지 판별하기 위해 시간적 움직임 벡터 (Temporal Motion Vector)가 사용되었다. 시간적 움직임 벡터는 참조 영상 (Reference Picture)의 움직임 벡터 (Motion Vector)로서 참조 영상 내의 화면간 부호화된 매크로블록마다 저장되고 차후 이 참조 영상을 이용하는 영상에서 사용할 수 있다. HEVC 에서는 시간적 움직임 벡터를 H.264/AVC 보다 직접적으로 움직임 벡터 예측과 움직임 병합에 이용하여 부호화 효율을 향상시키고 있다. 움직임 벡터 예측 시에 공간적으로 인접한 유닛 (Unit)의 움직임 벡터뿐만 아니라 시간적 움직임 벡터를 이용할 수 있다. 움직임 병합에서는 공간적으로 인접한 유닛 혹은 시간적으로 인접한 유닛의 움직임 벡터뿐만 아니라 참조 영상 색인 (Reference Picture Index), 화면간 예측 모드 (Inter Prediction Mode)를 병합 대상 유닛으로부터 유도하여 사용한다.

하지만 HD 급 이상의 고해상도 영상 같이 영상의 공간적 해상도가 증가할수록 이러한 영상을 부호화하는데 사용되는 움직임 벡터의 개수뿐만 아니라 움직임 벡터가 가질 수 있는 값의 범위도 증가한다. 결국 영상의 공간적 해상도가 증가할수록 시간적 움직임 벡터를 저장하는데 필요한 메모리 요구량이 증가하게 된다. 본 논문에서는 시간적 움직임 벡터를 저장하는데 필요한 메모리 요구량을 감소하기 위해서 시간적 움직임 벡터의 성분 값을 절삭한 뒤 메모리에 저장하는 방법을 제안한다. 이어지는 2 절에서는 종래 움직임 벡터 압축 방법에 대해 살펴보고, 3 절에서는 제안하는 방법을 설명하고, 4 절에서 제안하는 방법의 실험 결과를 분석한 뒤, 5 절에서 본 논문에 대한 결론을 맺는다.

표 1. 시간적 움직임 벡터를 저장하는데 필요한 메모리 크기

	가로 크기	세로 크기	x 성분의 다이내믹 레인지 (첫 부호화 유닛)	y 성분의 다이내믹 레인지 (첫 부호화 유닛)	x 성분의 다이내믹 레인지 (마지막 부호화 유닛)	y 성분의 다이내믹 레인지 (마지막 부호화 유닛)	x 성분의 비트 심도	y 성분의 비트 심도	영상 내 4x4 유닛의 개수	블록당 움직임 벡터 수	참조 영상 리스트의 수	참조 영상 리스트 당 참조 영상 수	필요한 메모리 요구량 (Mbits/장)
A 종류	2560	1600	−252 ~ 10236	−252 ~ 6396	−10460 ~ 28	−6620 ~ 28	15	14	256000	2	2	2	**59.39**
B 종류	1920	1080	−252 ~ 7676	−252 ~ 4316	−7900 ~ 28	−4540 ~ 28	14	14	129600	2	2	2	**29.03**
C 종류	832	480	−252 ~ 3324	−252 ~ 1916	−3548 ~ 28	−2140 ~ 28	13	12	24960	2	2	2	**4.99**
D 종류	416	240	−252 ~ 1660	−252 ~ 956	−1884 ~ 28	−1180 ~ 28	12	12	6240	2	2	2	**1.20**
E 종류	1280	720	−252 ~ 5116	−252 ~ 2876	−5340 ~ 28	−3100 ~ 28	14	13	57600	2	2	2	**12.44**
평균													**21.41**

2. 종래 움직임 벡터 저장 방법

HEVC 시험 모델 (HEVC Test Model: HM) 2.0 에서는 참조 영상의 움직임 벡터인 시간적 움직임 벡터를 향상된 움직임 벡터 예측과 움직임 병합에서 이용하여 여러 실험 조건에서 평균 약 2.2%의 BD (Bjøntegaard Delta)−Rate 감소를 얻을 수 있었다. 하지만 시간적 움직임 벡터를 사용하는 것은 참조 영상의 움직임 벡터를 메모리에 저장해야 하므로, 추가적인 메모리 요구량을 필요로 하게 된다. 제 4 차 JCT-VC 회의에서는 시간적 움직임 벡터로 인해 발생하는 메모리 저장공간 크기를 감소하기 위해 시간적 움직임 벡터의 공간적인 해상도를 감소하여 메모리에 저장하는 방법이 채택되었다 [4]. 기존의 HM 1.0 에서는 4x4 크기의 예측 유닛 (Prediction Unit)마다 모든 시간적 움직임 벡터가 그림 1 과 같이 저장되어야 했지만, [4]의 방법을 이용해서 그림 2 와 같이 16x16 영역 중 첫 번째 4x4 예측 유닛의 움직임 벡터가 대표 움직임 벡터로 결정되어, 대표 움직임 벡터만 메모리에 저장될 수 있다. 따라서 시간적 움직임 벡터의 메모리 저장공간뿐만 아니라 해당 메모리에 접근하는데 필요한 메모리 접근 대역폭 역시 1/16 으로 줄어들었다. 실질적으로 움직임 벡터뿐만 아니라 참조 영상 색인과 화면간 예측 모드 정보를 저장하는데 필요한 메모리 요구량도 1/16 으로 줄어들 수 있다.

표 1 은 시간적 움직임 벡터를 저장하는데 필요한 메모리의 크기를 나타내고 있다. 영상의 공간적 해상도가 커짐에 따라 메모리에 저장해야 할 움직임 벡터의 개수와 움직임 벡터가 가질 수 있는 값의 범위가 커지게 되므로, 시간적 움직임 벡터를 저장하는데 필요한 메모리 크기는 움직임 벡터의 공간적 해상도뿐만 아니라 움직임 벡터가 메모리에 저장될 때의 비트 심도에 의해 영향을 받는 것을 알 수 있다. 각 움직임 벡터의 다이내믹 레인지 (Dynamic Range)는 1/4 화소 단위로 계산되었으며, HEVC 표준화에서 사용하는 다양한 영상의 공간적 해상도에 따라 각각 계산되었다. 표 1 을 통해 알 수 있듯이 현재 실험 조건에서 최악의 경우, A 종류 영상의 시간적 움직임 벡터의 성분 값을 표현하는데 있어서, x 성분과 y 성분 각각 최대 15 비트와 14 비트가 필요하며, 참조 영상 1 장 당 약 59.39 Mbits 가 필요한 것을 알 수 있으며, 또한 5 개 영상 종류 평균 21.41 Mbits 가 필요한 것을 알 수 있다.

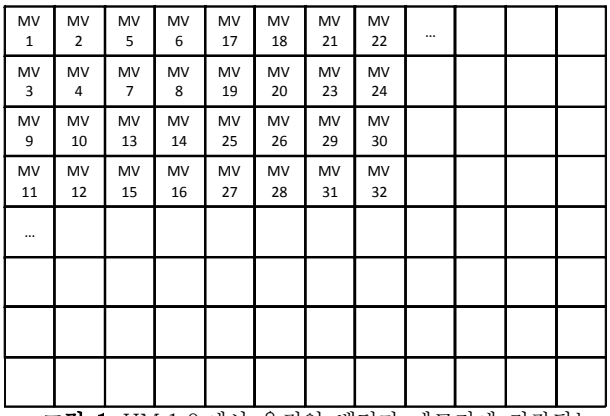

그림 1. HM 1.0 에서 움직임 벡터가 메모리에 저장되는 형태

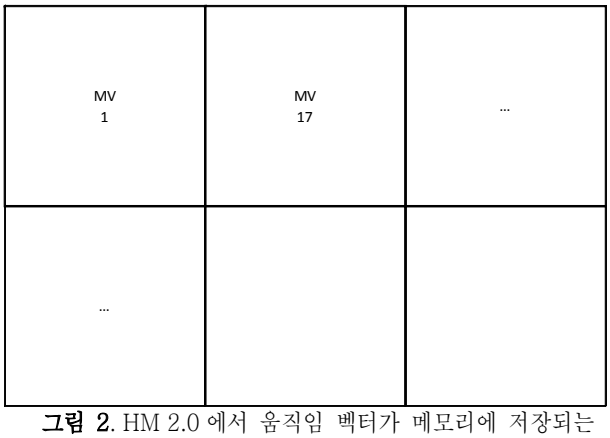

그림 2. HM 2.0 에서 움직임 벡터가 메모리에 저장되는 형태

따라서 시간적 움직임 벡터를 저장하는데 필요한 메모리 요구량을 줄이기 위해서 본 논문에서는 시간적 움직임 벡터의 각 성분 값을 절삭한 뒤 메모리에 저장하는 방법에 대해서 제안하며, 절삭되어 저장된 시간적 움직임 벡터는 차후 향상된 움직임 벡터 예측 및 움직임 병합에 이용된다.

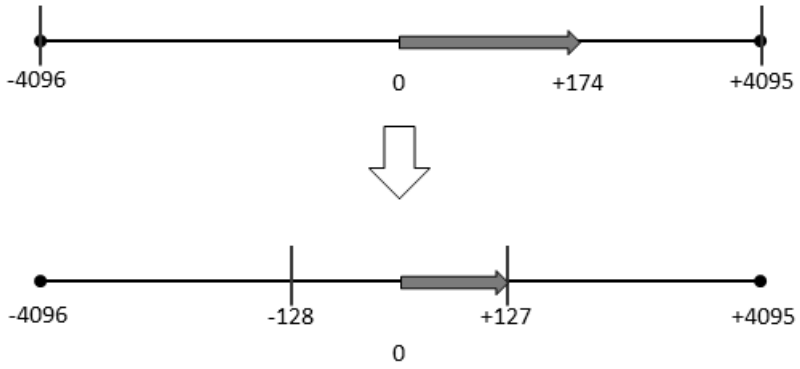

그림 3. 제안하는 시간적 움직임 벡터 절삭 방법

3. 제안하는 시간적 움직임 벡터 절삭 방법

본 논문에서는 각 시간적 움직임 벡터 성분의 최소값과 최대값을 제약하여 시간적 움직임 벡터의 비트 심도를 감소하는 방법에 대해서 제안한다. 제안하는 방법은 시간적 움직임 벡터를 저장하는데 필요한 비트 심도 정보에 따라 시간적 움직임 벡터가 메모리에 저장되기에 앞서 각 성분 값을 절삭한다. 비트 심도 정보는 각 영상의 움직임 특성을 반영하여 시퀀스 매개변수 집합 (Sequence Parameter Set)에서 전송될 수 있으며, 표준의 레벨 (Level)로 정의되어 특정 비트 심도로만 시간적 움직임 벡터가 저장되게 할 수 있다. 즉, 하나의 영상 전체에 대해서 시간적 움직임 벡터는 모두 고정 비트로 저장될 수 있다. 움직임 벡터의 각 성분은 식 (1)과 (2)를 이용해서 고정 N 비트로 절삭될 수 있다.

$$clippedMV\_X = \min(1<<(N-1)-1, \max(-1<<(N-1), MV\_X)). \quad (1)$$

$$clippedMV\_Y = \min(1<<(N-1)-1, \max(-1<<(N-1), MV\_Y)). \quad (2)$$

MV_X 와 *MV_Y* 는 움직임 벡터의 x 성분과 y 성분을 의미하며, 절삭된 움직임 벡터의 각 성분 값은 *clippedMV_X* 와 *clippedMV_Y* 이며, 메모리에 고정 N 비트로 저장될 수 있다. 그림 3 은 제안하는 움직임 벡터의 절삭 방법에 대한 예이다. 13 비트의 심도를 가지는 특정 움직임 벡터 성분 값인 +174 를 8 비트로 절삭할 경우 +127 로 되며, 이와 같은 방법으로 모든 움직임 벡터 성분 값은 8 비트로 메모리에 저장된다.

예를 들어 POC (Picture Order Count)가 10 인 영상을 먼저 부호화하고 POC 가 11 인 영상이 POC 가 10 인 영상을 참조하여 부호화한다고 가정하면, 제안하는 방법은 아래와 같은 순서로 수행된다.

1) POC 가 10 인 영상 내 부호화 유닛을 모두 부호화

2) POC 가 10 인 영상 내 전체 부호화 유닛의 움직임 벡터의 각 성분 값을 절삭하여 저장

3) POC 가 11 인 영상 내 부호화 유닛의 움직임 벡터 예측 및 움직임 병합을 수행할 때, POC 가 10 인 영상에서 절삭되어 저장된 움직임 벡터를 시간적 움직임 벡터로 이용

4) POC 가 11 인 영상 내 전체 부호화 유닛의 움직임 벡터를 절삭하여 저장

4. 실험 결과

제안하는 방법은 HM 2.0 스프트웨어에 구현하였고, 기본적인 실험 조건은 [5]에서 정의하고 있는 실험 조건 중 4 가지의 화면간 예측 구조인 고효율 임의접근 (Random Access), 고효율 저지연 (Low Delay), 저복잡 임의접근, 저복잡 저지연 실험 조건을 이용하였다.

표 3 과 4 는 제안하는 방법 중 고정 8 비트에 대한 실험 결과를 나타낸다. 표 3 은 고효율 임의접근 실험 조건의 결과이며, 17 개의 영상 평균 약 0.3%의 BD-Rate 손실을 보인다. 표 4 는 고효율 저지연 실험 조건의 결과이며, 16 개의 영상 평균을 보면 BD-Rate 손실이 없는 것을 알 수 있다. 고효율 임의접근 실험의 SteamLocomotive 영상의 경우 3.4%의 손실을 보이는데, 이는 영상의 공간적 해상도도 크고, 영상의 움직임도 다른 영상들 보다 상대적으로 크며, 임의접근 실험 조건 상 참조 영상과 부호화 대상 영상 간의 거리가 멀어지기 때문에 손실이 크게 발생한다는 것을 알 수 있다.

제안하는 방법을 4 가지 화면간 예측 구조 실험 조건에서 실험했을 때, 8 비트 방법은 평균 0.2% BD-Rate 손실, 6 비트 방법은 평균 0.8% BD-Rate 손실, 10 비트 방법은 평균 0.0% BD-Rate 손실을 보임을 알 수 있었다. 비록 부호화 효율에서 손실이 발생하더라도 표 2 와 같이 HM 2.0 대비 8 비트 방법은 약 43.28%의 메모리 요구량 감소, 6 비트 방법은 약 57.46%의 메모리 요구량 감소, 10 비트 방법은 약 29.10%의 메모리 요구량 감소를 보인다. HM 1.0 과 비교할 경우, [4]의 방법과 함께 이용하면 8 비트 방법은 3.55%의 메모리 크기가 필요하며, 6 비트 방법은 2.66%의 메모리 크기가 필요하며, 10 비트 방법은 4.43%의 메모리 크기만 필요하다.

5. 결론

본 논문에서는 시간적 움직임 벡터의 각 성분 값을 절삭하고 메모리에 저장하는 방법에 대해서 제안했다. 제안하는 방법은 비트 심도 정보에 따라 움직임 벡터의 최소값과 최대값을 제한하는 방법을 이용하였으며, 제안하는 8 비트 방법을 이용할 경우, HM 2.0 과 비교해서 0.2%의 부호화 효율 손실이 있지만 시간적 움직임 벡터를 저장하는데 필요한 메모리 요구량을 약 반 정도로 줄일 수 있었다.

표 2. 제안하는 방법을 이용했을 때 필요한 메모리 크기 (Mbits/장) 및 메모리 크기 감소율

	참조 샘플 저장공간 크기 (4:2:0, 8 비트)	필요한 메모리 크기				
		HM 1.0	HM 2.0	HM 2.0 + 제안하는 방법 (6 비트)	HM 2.0 + 제안하는 방법 (8 비트)	HM 2.0 + 제안하는 방법 (10 비트)
A 종류	196.61	59.39	3.71	1.54	2.05	2.56
B 종류	99.53	29.03	1.81	0.78	1.04	1.30
C 종류	19.17	4.99	0.31	0.15	0.20	0.25
D 종류	4.79	1.20	0.07	0.04	0.05	0.06
E 종류	44.24	12.44	0.78	0.35	0.46	0.58
평균	72.87	21.41	1.34	0.57	0.76	0.95
메모리 크기 감소율		100%	6.25%	2.66%	3.55%	4.43%

표 3. 고효율 임의접근 실험 조건에서 제안하는 방법의 결과 (8 비트)

영상		BD-Rate (%)
A 종류 (1600p)	Traffic	0.2
	PeopleOnStreet	0.0
	Nebuta	0.1
	SteamLocomotive	3.4
	평균	0.9
B 종류 (1080p)	Kimono	0.1
	ParkScene	0.2
	Cactus	0.1
	BasketballDrive	0.5
	BQTerrace	0.2
	평균	0.2
C 종류 (WVGA)	BasketballDrill	0.0
	BQMall	0.2
	PartyScene	0.0
	RaceHorses	0.2
	평균	0.1
D 종류 (WQVGA)	BasketballPass	0.1
	BQSquare	0.0
	BlowingBubbles	0.0
	RaceHorses	0.1
	평균	0.0
평균		**0.3**

표 4. 고효율 저지연 실험 조건에서 제안하는 방법의 결과 (8 비트)

영상		BD-Rate (%)
B 종류 (1080p)	Kimono	0.0
	ParkScene	0.0
	Cactus	0.1
	BasketballDrive	0.0
	BQTerrace	0.0
	평균	0.0
C 종류 (WVGA)	BasketballDrill	0.0
	BQMall	−0.1
	PartyScene	0.0
	RaceHorses	0.0
	평균	0.0
D 종류 (WQVGA)	BasketballPass	−0.2
	BQSquare	0.0
	BlowingBubbles	−0.1
	RaceHorses	0.0
	평균	−0.1
E 종류 (720p)	Vidyo1	0.1
	Vidyo3	0.1
	Vidyo4	−0.2
	평균	0.0
평균		**0.0**

감사의 글

본 연구는 방송통신위원회의 한국전자통신연구원 연구개발지원사업 "무안경 다시점 3D 지원 UHDTV 방송 기술 개발"의 연구결과로 수행되었음"(KCA-2011-11921-02001))

참고 문헌

[1] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for Generic Audiovisual Services," ITU-T Recommendation H.264 and ISO/IEC 14496-10, Sep. 2008.

[2] ITU-T VCEG and ISO/IEC MPEG, "Joint Call for Proposals on Video Compression Technology," document VCEG-AM91 of VCEG and N11113 of MPEG, Jan. 2010.

[3] T. Wiegand, B. Bross, W.-J. Han, J.-R. Ohm, and G. J. Sullivan, "WD3: Working Draft 3 of High-Efficiency Video Coding," document JCTVC-E603 of JCT-VC, Mar. 2011.

[4] Yeping Su and Andrew Segall, "CE9: Reduced resolution storage of motion vector data," document JCTVC-D072 of JCT-VC, Jan. 2011.

[5] Frank Bossen, "Common test conditions and software reference configurations," document JCTVC-D600 of JCT-VC, Jan. 2011.